НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

.

Дипломна робота

на тему: система оцінювання ідентичності текстів з урахуванням синонімії.

Студент групи ТР-51 Синельник Олександр Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник роботи доц., к.т.н. Лабжинський В. А. _____
(вчені ступінь та звання, прізвище, ініціали) (підпис)

Кількість сторінок _____

Кількість ілюстрацій _____

Київ – 2019

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О. В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050101 “ Комп’ютерні науки’’

на тему: Система оцінювання ідентичності текстів з урахуванням синонімії

Виконав: студент 4 курсу, групи ТР-51

Синельник Олександр Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, доц., к.т.н., Лабжинський В. А. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент Старший науковий співробітник інституту «ІПРІ НАН України», к.т.н. Сенченко В. Р. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “Комп’ютерні науки’’

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О. В. Коваль
(підпис)
”___”_____2019 р.

ЗАВДАННЯ

на дипломну роботу студенту

Синельнику Олександру Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “ Система оцінювання ідентичності текстів з урахуванням синонімії ”
керівник робот Лабжинський Володимир Анатолійович, доц., к.т.н., доцент
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”___”_____ 201__ р. № _____

2. Строк подання студентом роботи ___ _____ 201__ р.
3. Вихідні дані до роботи текстовий файл з розширенням .txt, в якому зберігаються результати оцінювання ідентичності текстів з урахуванням синонімії
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення та засоби оцінювання ідентичності текстів, спроектувати архітектуру системи оцінювання ідентичності текстів з урахуванням синонімії, розробити програмне забезпечення, розробити інтерфейс користувача
5. Перелік ілюстраційного матеріалу (з точним зазначенням обов’язкових креслень) 1. Мета та завдання роботи 2. Актуальність теми 3. Огляд існуючих рішень 4. Сценарій використання системи 5. Функції системи оцінювання ідентичності текстів з урахуванням синонімії 6. Формули розрахунку 7. Використані програмні засоби 8. Висновки
Дата видачі завдання ”___”_____ 201__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	14.10.2018-23.12.2018	
2.	Розробка архітектури та загальної структури системи	2.02.2019-3.03.2019	
3.	Розробка структур окремих підсистем	4.03.2019-14.04.2019	
4.	Підготовка матеріалів	15.04.2019-18.04.2019	
5.	Програмна реалізація системи	18.04.2019-14.05.2019	
6.	Захист програмного продукту	15.05.2019	
7.	Оформлення пояснювальної записки	16.05.2019-3.06.2019	
8.	Передзахист	28.05.2019	
9.	Захист	17.06.2019-22.06.2019	

Студент _____ Синельник О. С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Лабжинський В. А.
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

У навчальних закладах багато курсових та лабораторних робіт студентів складаються з описової (текстової) і програмної частини. Тому актуальним є створення програм, що дозволяють виявити списування тексту. У дипломній роботі зроблено акцент не стільки на розвиток конкретного методу пошуку плагіату або на розвиток засобів автоматизації такого пошуку при великому обсязі бази даних, скільки на покращеня пошуку ідентичності текстів завдяки урахуванню синонімії. У наш час досить широко використовується запозичення текстів з заміною багатьох слів на синоніми, та сучасні системи пошуку плагіату не здійснюють оцінку текстів з урахуванням синонімії. Актуальність досліджуваної проблеми підтверджується чималим числом публікацій, в яких розглядаються різні підходи, методи реалізації і конкретні інструментальні засоби пошуку плагіату. Новизна роботи полягає в тому, що в ній об'єднані в рамках єдиних інструментальних засобів як алгоритми пошуку плагіату в довільних текстах з урахуванням синонімії.

В роботі розглядаються в теоретичному і практичному плані такі питання:

- методи аналізу довільних текстів з точки зору наявності ідентичних фрагментів та фрагментів, замінених синонімами;
- розробка набору інструментів для відсіювання стоп-слів та службових символів з текстів для їх коректного аналізу;
- реалізація алгоритму пошуку запозичених фрагментів у текстах з урахуванням синонімії.

Все це в сукупності дозволяє значно покращити наявні методи пошуку ідентичних фрагментів і виконати більш глибокий аналіз порівнюваних текстів.

1 ОГЛЯД ОЦІНЮВАННЯ ІДЕНТИЧНОСТІ ТЕКСТІВ З УРАХУВАННЯМ СИНОНІМІЇ

1.1 Поняття оцінювання ідентичності текстів з урахуванням синонімії

Лексика – це головна складова частина і вся сукупність слів мови, її словниковий склад. Системність є однією з найважливіших характеристик мови і основним предметом вивчення лінгвістики. Як і інші рівні мови, лексика є системою. Це означає, що слова певним чином пов'язані один з одним. Як зазначає І.П. Слєсарєва, системність мови є однією з основоположних проблем в лінгвістиці [31].

Під системою в загальному сенсі розуміється сукупність елементів, пов'язаних внутрішніми відносинами. За В. І. Половниковою, під системністю лексики розуміється своєрідність і різноманіття типів угрупування лексичних одиниць, їх смислові відносини, а також характер пов'язаності і взаємодії один з одним. При такому розумінні лексична система – це внутрішньо організована сукупність мовних елементів, закономірно пов'язаних між собою відносно стійкими відносинами і постійно взаємодіючих. Власне мовна система визначається як множина мовних елементів, що знаходяться в закономірних зв'язках і відносинах один з одним, що характеризується певною цілісністю [32].

Говорячи про основні проблеми опису лексики в аспекті системності, Е. І. Зінов'єва наводить думку Ю. С. Сорокіна про конкретний прояв системності в лексиці, який пише: «Система в лексиці проявляється: 1) в семантичній структурі слова, що відбиває його ставлення до дійсності; 2) в слово відносинах, зв'язках слів з іншими словами за своєю формою, стосовно слова до тієї чи іншої загальної або приватної лексико-граматичної категорії (частини мови, гніздо слів); 3) в семантичних зв'язках і відносинах слова з іншими словами: синонімія, антонімія;

об'єднань слів в лексико-тематичні групи, окремі термінологічні ряди, семантичні мікросистеми; 4) в контекстуальних, семантико-фразеологічних зв'язках слів: наявності стійкої сполучуваності слів у мові, визначених однакових типів цієї сполучуваності; 5) в об'єднанні, групуванні слів по стилістичним властивостям [33].

Словниковий склад мови розглядається як система лексичних одиниць, пов'язаних один з одним через приналежність до певних класів слів. Лексична система – це система взаємопов'язаних класів слів, що перетинаються, різного обсягу та характеру. Найбільш типовим проявом системності лексики визнають семантичні угрупування слів, для позначення яких використовуються різні терміни: лексико-семантичні групи, тематичні групи, семантичні парадигми, семантичні, асоціативні та понятійні поля, функціональні класи тощо. Найбільш вживаним і отримав достатньо визнання термін лексико-семантична група.

Системність лексики є об'єктивним фактом, оскільки під системою розуміється множина взаємопов'язаних і взаємообумовлених елементів. У лексичній системі мови виділяють групи слів, пов'язаних спільністю або протилежністю значення – синоніми і антоніми; подібних або протиставлених за стилістичними властивостям; об'єднаних загальним типом словотвору; пов'язаних спільністю походження тощо.

Прояв системності в лексико-семантичній області мови полягає в здатності слів об'єднуватися один з одним за змістом в різні лексико-семантичні класи. Лексичні групи можуть формуватися і на чисто лінгвістичних підставах. Наприклад, лінгвістичні особливості слів дозволяють згрупувати їх в частини мови за лексико-семантичними і граматичними ознаками.

Як відомо, існує три основних типи відносин, в яких виявляються системні відношення слів: парадигматичні, синтагматичні та деривaційні.

Системні відносини в групах слів, які об'єднані спільністю ознак, називаються парадигматичними. До парадигматики відносять угрупування слів в системі мови, основою яких виступає опозиція – синонімія, антонімія, гіпонімія, паронімія, гніздо слів, сім'я слів, лексико-семантична група, а також найбільш загальне угрупування

слів – поле. У лексико-семантичну парадигму об'єднуються слова за деякою семантичною ознакою (наприклад, за ознакою «професія»: учитель, лікар, льотчик; за ознакою «величина»: великий, маленький тощо). В лексичні парадигми об'єднують синоніми, антоніми, слова однієї тематичної групи (день – ніч – ранок – вечір).

Під синтагматикою розуміється сукупність правил і закономірностей, що визначають відносини між одиницями в мовного ланцюга – синтагматичні відносини, які протиставляються парадигматичним або асоціативним відношенням послідовності. До синтагматики відносять угрупування слів за їх розташуванням в мові відносно один одного (сполучуваність, аранжування). Синтагматичні відносини виявляють здатність слів з'єднуватися один з одним, тобто сполучуваність, яка визначається предметно-змістовими зв'язками, граматичними властивостями, лексичними особливостями слів. Особливості сполучуваності окремих слів в значній мірі залежать від контексту, тому синтагматичні зв'язки в більшій мірі, ніж парадигматичні, схильні до змін, обумовленим змістом промови.

Деривaційні (словотвірні) відносини – відносини смислової мотивації одних слів іншими: ліс-лісник, вода́-во́да. У процесах деривації відбувається зміна форми і семантики одиниць, прийнятих за вихідні.

Таким чином, при лінгвістичному аналізі мовних явищ, а також при лінгводидактичному описі мови як іноземної, слід усвідомлювати важливу роль системності мови і виявляти системні відносини в лексиці з урахуванням взаємозв'язку парадигматичних, синтагматичних і деривaційних відносин. Це пов'язано з тим, що вивчення лексики з опорою на формування різних зв'язків слів підвищує усвідомленість і зменшує механістичність в засвоєнні лексичних одиниць.

Системність лексики можна простежити в наступних групах слів: 1) семантичне поле, 2) лексико-семантична група, 3) тематична група, 4) ситуативна група, 5) комунікативна група, 6) родо-видова група, 7) синонімічний ряд , 8) антонімічна пара, 9) словотворче гніздо, 10) епідігматична група.

З основних виділяються дослідниками словникових об'єднань російської мови об'єктом пильної уваги в цій роботі є лексико-семантичне поле і лексико-семантична група.

У сучасній лінгвістиці об'єктом дослідження є мовні об'єднання різних видів полів: семантичні, лексико-семантичні, функціонально-семантичні, концептуальні, фразеосемантичних, лінгвокультурологічні поля тощо. Поле стало однією з найважливіших категорій лінгвістики. Це поняття використовується в лексикології, в граматиці, у фразеології, в лінгвокультурології. Це відбувається завдяки тому, що поле об'єднує в ієрархічну систему одиниці, які мають загальне значення.

На думку Е. І. Зінов'євої, термін поле виступає в лінгвістиці як родове поняття, але в більшій мірі застосовується до одиниць лексичного рівня мови. Наприклад, визначення в лінгвістичному енциклопедичному словнику: поле – сукупність мовних (головним чином лексичних) одиниць, об'єднаних спільністю змісту (іноді також спільністю формальних показників), що відображають понятійну, предметну чи функціональну схожість явищ [34].

Одним із завдань сучасних семантичних досліджень є вивчення значень одиниць мови в їх всіляких зв'язках, вивчення складу і структури різних семантичних полів. Лінгвісти вважають, що аналіз мовного змісту методом поля є найбільш ефективним і перспективним. Основний сенс методу семантичного поля полягає в розумінні лексики як системи, виразом якої є різні угрупування слів. Лексичний склад мови представляється у вигляді різних об'єднань. Лексико-семантичне поле грає важливу роль у вивченні семантики лексичних одиниць.

Сучасними уявленнями філологів лексико-семантичне поле – це сукупність лексем, що позначають певне поняття в широкому сенсі цього слова, яка включає до свого складу слова різних частин мови. Лексико-семантичне поле характеризується рядом ознак системності як в синхронному плані (семантична співвідносність лексем, «ділять» поле між собою, наявність гіпонімії і гиперонімів), так і в генетично-діахронічному плані (певний набір неодноразово реалізованих мотиваційних

моделей, повторюваність словотворчих моделей, повторюваність етимологічних гнізд, що породжують лексику поля).

У своєму дослідженні під лексико-семантичним полем ми розуміємо ієрархічну структуру множини лексичних одиниць, що об'єднуються загальним (інваріантним) значенням і відбивають у мові певну понятійну сферу. Є. І. Зінов'єва справедливо вважає, що лексико-семантичне поле (далі – ЛСП) об'єднує вербальний ряд одиниць – це слова і словосполучення, об'єднані на основі спільності [34].

Організаційним центром лексико-семантичного поля часто є концепт, який, як правило, носить те ж ім'я, що і назва поля. Концепт є змістом вербального знака, але часто не одиничного слова, а реалізується саме в усій лексико-семантичній парадигмі (в поле), має понятійну, образну і прагматичну складові, які можна виявити через аналіз парадигматичних, синтагматичних і деривaційних зв'язків одиниць, що входять в ЛСП.

Для лексико-семантичного поля постулюється наявність загальної (інтегральної) семантичної ознаки, що об'єднує всі одиниці поля і зазвичай виражається лексемою з узагальненим значенням (архісема), наприклад, ознака «переміщення в просторі» в семантичному полі дієслів руху: «йти», «бігти», «їхати», «плисти», «летіти» тощо, і наявність часткових ознак (від однієї і більше), за якими одиниці поля відрізняються один від одного, наприклад, «швидкість», «спосіб», «середовище пересування» тощо.

За словами Є. І. Зінов'євої, ЛСП досліджується як частина мовної системи з потенційно властивими їй на рівні мови системними зв'язками. При цьому аналізується структура поля та характер семантичних відносин між елементами. Основним матеріалом для дослідження в даному випадку є дефініції тлумачних словників мови, а в якості ілюстративного матеріалу можуть залучатися цитати з текстів, концентруючи словникове значення одиниць ЛСП і їх парадигматичні зв'язки і демонструючи синтагматичні відношення елементів поля [34].

Кожне ЛСП є складною парадигматичною структурою. Важливою властивістю поля визнається наявність зв'язків між його елементами.

Парадигматичні системні відносини одиниць поля проявляються в семантичних опозиціях. Під опозицією розуміється протиставлення однорідних одиниць, яке семантично істотно в мові. Кожна одиниця в системі поля може включатися в цілий ряд опозицій, в результаті чого створюються цілі парадигматичні угрупування. Виявлення структурних відносин між елементами поля – одне з головних завдань будь-якого дослідження семантики.

Лексико-семантичне поле розглядається як об'єднання лексичних одиниць, яке складається з цілого ряду лексико-семантичних груп, тобто ЛСП і ЛСГ співвідносяться як загальне і часткове. Семантичне поле розуміється як вищий рівень лексико-семантичних зв'язків парадигматичного характеру і визнається більший радіус дії лексичних одиниць в полі.

Оскільки одним з головних завдань нашого дослідження є класифікація і аналіз відібраних одиниць в рамках лексико-семантичних груп, перейдемо до більш докладного розгляду поняття лексико-семантичної групи.

Лексичні синоніми – це слова різні за звучанням і написанням, але близькі або тотожні за значенням. У термінній лексиці простежуємо функціонування таких пар серед лексичних синонімів:

- термін чужомовного походження – особливий український відповідник:
- позичений термін та самобутній український термін: абсорбер-вбирач;
- термін, що складається з чужомовних компонентів, – особливий український аналог: анемометр-вітромір;
- гібридний термін (гібридними термінами вважатимемо слова, які мають чужомовну твірну основу або лише корінь) – самобутній український еквівалент: ажурний-прозірчастий;
- синонімічні українські терміни;

- синонімічні терміни, позичені з різних мов (алмаз-діамант, дисплей-монітор);
- терміни, позичені з однієї мови (азот-нітроген, манган-марганець).

Синонімами термінів можуть виступати застарілі (вата-бавна, патронаш-ладівниця) та діалектні слова (розчин-чамур). Цікаво, що таке паралельне вживання фіксують не лише термінологічні словники 20-30-х рр., але й деякі сучасні. Застарілі та діалектні слова в термінології не варто, на наш погляд, розглядати як нормативні назви наукових понять.

На граматичному рівні синонімія термінів виявляється в двох різновидах.

Словотвірні синоніми – це слова близькі або тотожні за значенням, які мають відмінності на рівні словотворення. А саме:

- омоосновні (одноосновні), в яких до однакових твірних основ приєднуються різні щодо звукового складу суфікси: а) два власне українські терміни: (випуск-випускання, килимар-килимник, остудник-остуджувач, мішковий-мішечний); б) два гібридні терміни (аквамаринний-аквамариновий, фініфтевий-фініфтяний).
- відображена або спадкова словотвірна синонімія – слова, утворені від словотвірних синонімів за допомогою тотожних або відмінних словотворчих афіксів: бетонництво-бетонярство, поруватість-пористість.
- гетероосновні словотвірні синоніми, в яких відмінними є як твірні основи, так і словотворчі суфікси (казаняр-котельник, обробник-оздоблювач) або лише твірні основи (барвність-колірність, охолодник-остудник).

Синтаксичні або структурні синоніми традиційно охоплюють як власне синтаксичні синоніми (словосполуки), так і синонімічні назви різної структури. Останні можна ще назвати різнорівневими синонімами, адже паралельно функціонують одиниці лексичного та синтаксичного рівнів. Це, зокрема, такі взаємозамінні назви: одноосновне слово – словосполука (дробарка-машина дробильна), складне слово чи словосполука (солодомлин-млин солодовий),

абревіатура-словосполука (ККД-коефіцієнт корисної дії, будмайданчик-будівельний майданчик).

Щодо власне синтаксичних синонімів, то серед них переважають паралельні аналітичні назви з однаковою кількістю компонентів (вихрові струми-струми Фуко, інтервал Найквіста-інтервал дискретизації), а також словосполуки з різною кількістю елементів, які ще називають квантитативними синонімами (галогенідсрібна голограма-голограма на галогенідсрібному носії, призма Амічі-призма прямого зору). Зрідка фіксуємо паралельне вживання синонімних складених термінів із різноструктурними атрибутивними елементами (машина кінна-машина коновідна, машина фанерна-машина фанеропильна) та одноструктурними (складними) атрибутивними елементами, які містять різнозвучні синонімічні основи (машина шерсточесальна-машина вовночесальна, машина дроворубна-машина дровокольна).

Різновиди варіантних термінів:

- фонетичні варіанти – звукові різновиди терміну, що не порушують принципу тотожності його словотвірної структури, лексичного і граматичного значень: гіроскоп-жироскоп, проміжок-промежок;
- акцентуаційні варіанти – різновиди того самого терміну, що різняться місцем наголосу: діóптрія-діоптрі́я;
- словотвірні варіанти – це спільнокореневі похідні, утворені за допомогою варіантних суфіксів: гартівник-гартувальник, зливний-зливальний;
- морфологічні варіанти – це слова, що характеризуються варіантністю граматичних категорій на рівні роду або числа: желатин-желатина, парафін-парафіну; перли-перла, козли-козла;
- синтаксичні варіанти – це словосполуки, що різняться одне від одного або порядком розташування компонентів, або заміною одних форм вираження синтаксичних функцій іншими: польовий ефект-ефект поля, холлівський помножувач-помножувач Холла.

Особливостями синонімії та варіантності в термінній лексиці є наявність значної кількості таких пар: позичені терміни – власне українські відповідники, терміни, утворені з позичених терміноелементів, й українські кальки, різноструктурні синоніми, терміни-епоніми та їхні відповідники, синоніми на знаковому рівні.

1.2 Огляд наявних систем пошуку плагіату

Практичним оцінюванням ідентичності текстів є перевірка на плагіат.

Плагіат – умисне привласнення авторства на чужий твір літератури, науки, мистецтва, винахід (повністю або частково). За порушення авторських прав передбачається кримінальна та цивільна відповідальність.

Авторство – приналежність твору автору.

Автор – творець будь-якого твору.

В останні роки з'являються нові види авторських творів, включаючи електронні, які раніше в законі "Про авторське право і суміжні права" відображені не були. Крім того, обсяг інформації з будь-якого виду творів та їх тематиці настільки великий, що в повному обсязі її проаналізувати неможливо ні автору чергового нового твору, ні експертам, тим більше, що багато авторів мимоволі «дублюють» один одного, так як приходять до одних і тих самих висновків, рішень і відкриттів абсолютно самостійно і незалежно від інших.

Плагіат також йде «в ногу з часом» і набуває все більш і більш витончені форми. Тому захист авторства і боротьба з плагіатом стає надзвичайно важким завданням. І, незважаючи на те, що за порушення закону "Про авторське право і суміжні права" існує кримінальна і цивільна відповідальність, притягнути плагіатора до тієї чи іншої формі відповідальності в багатьох випадках не є можливим, тому що все важче встановити (і довести) присутність або відсутність умисного привласнення чужих авторських прав.

Основні види плагіату

Є множина видів умисного плагіату (офіційно караного). Однак, поряд з умисним плагіатом, існує ненавмисний «плагіат» і завуальований «плагіат», які формально плагіатом не є і, природно, ніякому цивільному і кримінальному покаранню не підлягають. Але відокремити умисний плагіат від завуальованого і ненавмисного «плагіату» іноді буває просто неможливо.

Навмисний плагіат

Навмисний плагіат може бути не тільки авторським плагіатом, що полягає в умисному привласненні авторства на чужий твір (ціле або частини), а й рекламним плагіатом, і навіть диверсійним плагіатом. Рекламний і диверсійний плагіат найбільш часто зустрічається в Інтернет.

Рекламний плагіат виражається в тому, що рекламні плагіатори (з метою залучення покупців і / або відвідувачів) як заголовків і / або ключових слів навмисне використовують популярні імена і / або назви, які не мають нічого спільного з сайтом плагіатора.

Диверсійний плагіат також є умисним плагіатом. Він має таку ж мету, що і рекламний, але, крім того, може використовувати заголовки і ключові слова, імена людей і назви тих сайтів, які є антиподами сайту-плагіатора, а тому здатні уявити відомих людей і популярні сайти в абсолютно невірному світлі. Деякі плагіатори використовують спеціальні прийоми, які дозволяють (після виявлення обману) піти з сайту-плагіатора. Диверсійні сайти можуть навіть завдати шкоди комп'ютеру ту людину, яку вони заманили на свій сайт обманним шляхом.

Ненавмисний «плагіат»

Ненавмисний «плагіат» (наслідування, запозичення, випадкова схожість, збіг ідей або відкриттів, висловлених та / або утворених авторами незалежно один від одного, і т. п.) в тій чи іншій мірі властивий будь-якому, навіть зовсім новому, твору незалежно від його виду і теми.

Взагалі-то все нове, створене тим чи іншим автором, так чи інакше, засноване

на знаннях і досвіді багатьох попередників, які, хоча формально співавторами і не вважаються, але фактично ними є. І це було б пам'ятати тим, хто схильний вважати себе першовідкривачем, описуючи від свого імені вже давним-давно винайдений іншими «велосипед», і, не роблячи відповідних посилань, а як би намагаючись претендувати на авторство того, що фактично зроблено іншими, а ними просто переказується.

Ненавмисний «плагіат» може бути підсвідомим, випадковим (мимовільним) і «тиражованим».

Підсвідомий «плагіат» проявляється в тому, що людина, перечитавши множина інформації і прийшла, завдяки цьому, до тих чи інших рішень і висновків, включаючи, можливо, і нові, може використовувати при їх викладі то (з перечитаних нею робіт інших авторів), що найбільш повно збігається з його особистим баченням проблеми. При цьому вона може не пам'ятати і / або не розуміти, що фактично привласнює собі авторство на те, що спочатку йому не належить, і щиро вважати, що до цих рішень і висновків дійшла абсолютно самостійно.

Випадковий «плагіат» (мимовільний), який в науковій літературі зустрічається досить часто, полягає в тому, що до одних і тих самих висновків, рішень і навіть винаходів можуть самостійно прийти різні люди, причому абсолютно незалежно один від одного, і навіть викладати їх вони можуть майже однаковим чином.

«Тиражовані плагіат» отримав майже масове поширення, особливо в науковій літературі. Він виник через те, що посилатися прийнято (і зручно) на те джерело інформації, яке є найбільш популярним і видано найбільшим тиражем. При цьому майже ніхто з посилаються не задається питанням, чи є дана монографія істинним першоджерелом того конкретного матеріалу, на який посилаються і / або який запозичується. І хоча будь-яка солідна монографія, яка використовує, як правило, матеріали багатьох інших (більш дрібних) авторських робіт (менш популярних і менш затребуваних через свого малого тиражу і вузькості тематики), дає на них відповідні посилання, але автори наступних книг посилаються зазвичай на монографію, а не на

наведені в ній першоджерела.

Оскільки цікаві та корисні матеріали зазвичай багаторазово «тиражуються», переходячи з однієї монографії в іншу з посиланням на одну з попередніх монографій, то ім'я справжнього автора того чи іншого конкретного матеріалу, виявляється, найчастіше, в повному забутті. І хоча таке багаторазове «тиражування», причому найбільш цікавих і корисних матеріалів і навіть винаходів, формально плагіатом не рахується, але фактично їм все ж є.

Завуальований «плагіат»

Завуальований «плагіат» найбільш часто використовується в науково-популярній та науковій літературі. Він виражається в тому, що вже відоме (або навіть загальновідоме) викладається (навмисно чи ненавмисно) без відповідних посилань і підноситься з претензією на початкове авторство, яка часто підкреслюється забороною на використання будь-частини даної книги без письмового дозволу автора.

Спершу розглянемо онлайн-сервіси з пошуку фрагментів текстів на плагіат. Вони мають одну велику перевагу перед іншими сервісами, а саме те, що вони не вимагають установки на пристрої.

Text.ru

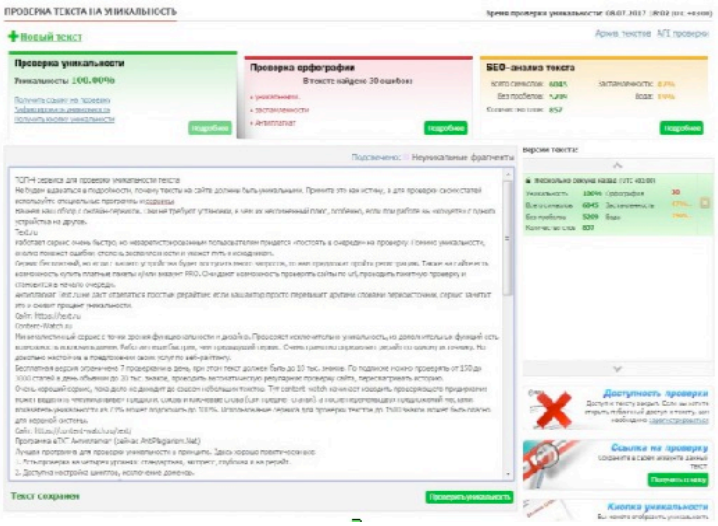

Рисунок 1.1 – Система пошуку плагіату Text.ru

Сервіс працює швидко, однак для незареєстрованих користувачів є один недолік: для перевірки їх тексту вони мають чекати своєї черги. Аналіз тексту, окрім унікальності, здатен виявити ступінь засміченості, помилки в словах та адреси до оригінальних текстів.

Сервіс є умовно безкоштовним. Та якщо ви будете користуватися їм дуже часто, сервіс запропонує вам пройти реєстрацію. Сервіс також передбачає придбання платних пакетів для користувача або платний обліковий запис, який дає змогу перевіряти сайти по url, проводити пакетну перевірку та проходить аналіз повз черги.

Антиплагіат Text.ru не дасть відбутися простим запозиченням: якщо ваш автор просто перепише іншими словами першоджерело, сервіс також врахує це як плагіат та зменшить відсоток унікальності тексту.

Content-Watch.ru

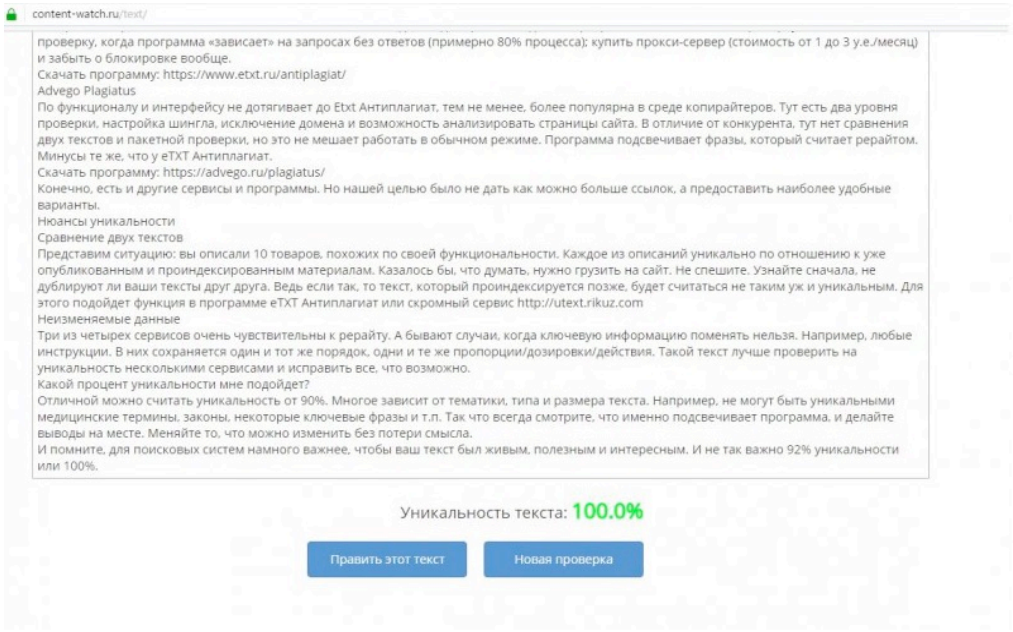

Рисунок 1.2 – Система пошуку плагіату Content-Watch.ru

Цей сервіс позиціонує себе як мінімалістичний з точки зору дизайну, а також функціональності. Цей сервіс працює значно швидше, аніж попередній, але ця швидкість обумовлена тим, що він перевіряє вхідні дані виключно на унікальність.

Нажаль, безкоштовна версія сервісу має досить великі обмеження, а саме:

- 7 перевірок в день;
- текст до 10 тисяч знаків.

Платна версія дозволяє перевіряти 150–3000 статей в день з обмеженим обсягом символів до 20 тисяч. Користувач зможе проводити регулярну перевірку сайту, а також переглядати власну історію перевірок на плагіат.

Сервіс передбачений на малі та середні обсяги інформації, але не здатен перевіряти на плагіат велику кількість інформації. Також сервіс не здатен перевіряти замінені місцями слова, тобто може підняти унікальність від 73 відсотків до 100 лише завдяки зміні послідовності слів у тексті. Сервіс не актуальний для дуже малих текстів, оскільки дуже часто встановлює велику кількість плагіату.

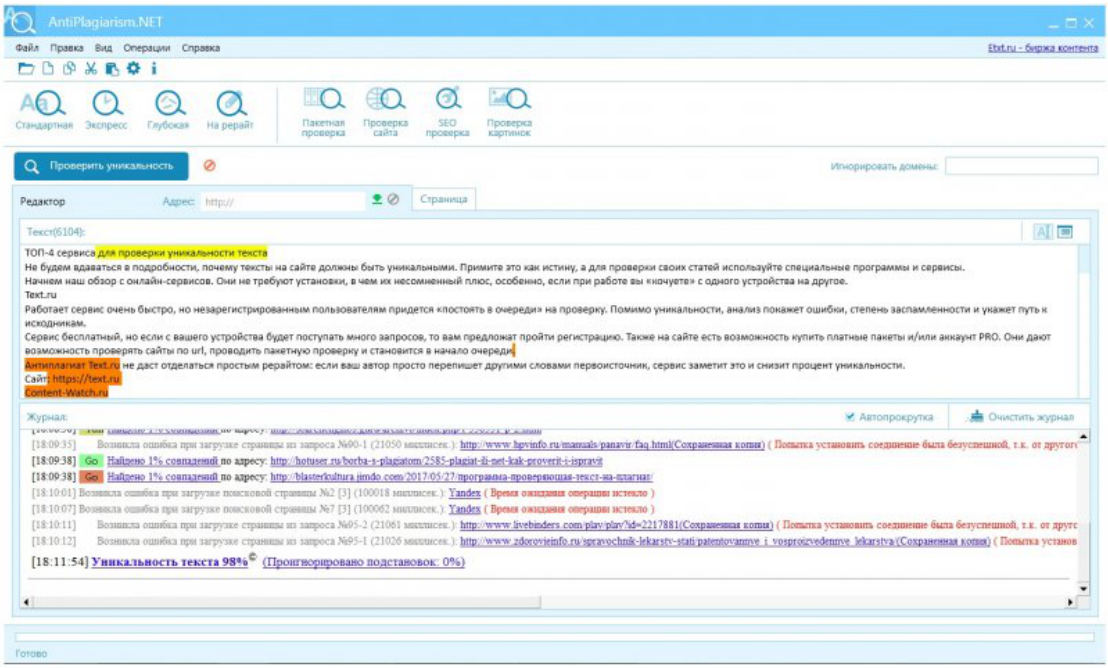

Рисунок 1.3 – Система пошуку плагіату eTXT Антиплагіат

Це одна з кращих програм для перевірки унікальності тексту. Серед переваг можна зазначити наступні:

- присутня градаційна перевірка текстів на чотирьох рівнях, а саме: стандартна, експрес, глибока і на запозичення фрагментів текстів;
- програма є гнучкою, користувач може налаштувати її під свої потреби;

- є можливість аналізу сайту та / або пакету документів;
- тільки за допомогою цієї програми є можливість перевірки двох локальних текстів, перевірки на унікальність зображень і завантаження кодів сторінок;
- програма має привабливий інтерфейс, що надає задоволення під час користування.

Нажаль, програма не є ідеальною, оскільки вона теж має власні мінуси, але їх важко уникнути завдяки специфіці поставлених задач.

В першу чергу, швидкість досить нижче, ніж в онлайн–сервісах, оскільки онлайн–сервіси мають найсучасніше обладнання та великі потужності машин, на яких відбуваються процеси перевірки та пошуку плагіату. Також програма все частіше просить користувача вводити коди безпеки, якщо було підвищено рівень перевірки.

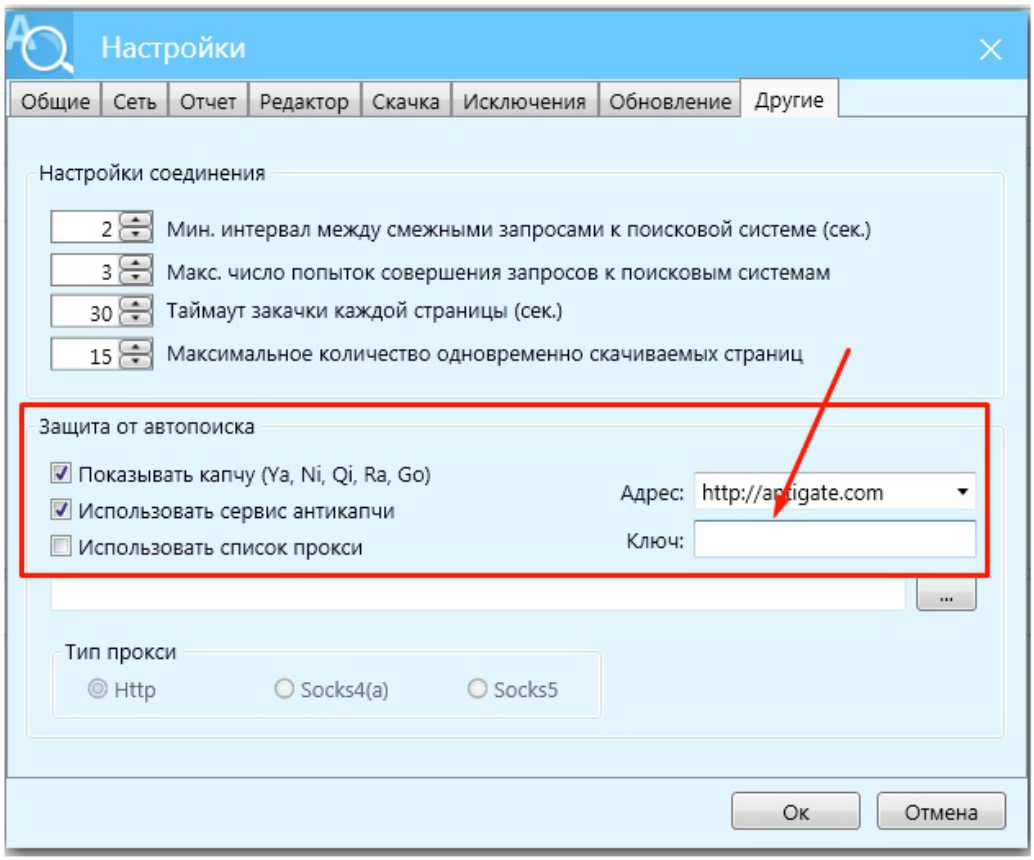

Рисунок 1.4 – Меню налаштування системи eTXT Антиплагіат

Останньою буде розглянуто систему Advego Plagiatus.

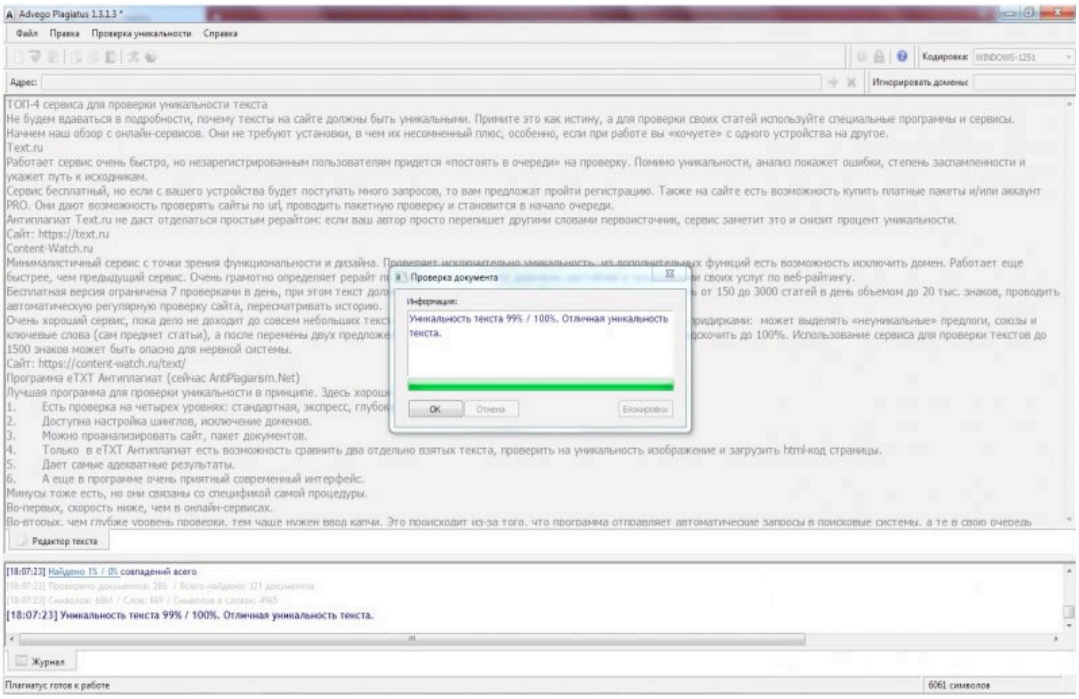

Рисунок 1.5 – Аналог розроблюваної системи Advego Plagiatus

Звичайно, що ця програма має не такий привабливий інтерфейс, що і eTXT Антиплагіат, та й функціональні можливості також дещо менший, проте вона популярніша серед авторів. Програма має два рівні перевірки: перевірка текстів та перевірка сторінок сайтів.

Мінуси Advego Plagiatus такі ж самі, що і у eTXT Антиплагіат, а також відсутність пакетної перевірки та порівняння двох текстів. Але відсутність цього функціоналу не заважає користуватися цією програмою.

Є дуже велика кількість інших сервісів для перевірки текстів на плагіат, проте вище перелічені є найбільш функціональними та зручними для користувачів. Користувач може обирати системи пошуку плагіату багатьма критеріями: між онлайн–сервісами та програмами, які потрібно встановлювати на власні пристрої, з можливістю пошуку без доступу до інтернету та навпаки, для малих чи великих обсягів інформації тощо.

2 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Особливості розробки

Розроблений додаток для оцінювання ідентичності текстів з урахуванням синонімії складається з програмного продукту, написаного мовою програмування Java. Він використовується для обрахунку відсотку запозиченості текстів, аналізу ідентичних фрагментів текстів з урахуванням синонімії. Середовищем розробки програмного продукту було обрано IntelliJ IDEA та інструментом для керування проектом – Maven.

2.2 Обґрунтування використання обраних засобів

2.2.1 Мова програмування Java

Мова програмування Java – це імпераивна мова високого рівня. На відміну від мов низького рівня, які нагадують машинний код, мови високого рівня повинні бути перетворені за допомогою компіляторів або інтерпретаторів. Це спрощує розробку, роблячи мову легшою для написання, читання і підтримування. Інша характеристика, яка стала причиною вибору мови Java для розробки додатку, є те, що це мова об'єктно-орієнтованого програмування. Розробка додатків ООП набагато простіше, а також допомагає підтримувати модульну, гнучку та розширювану систему.

Знання про ключові поняття ООП, такі як абстракція, інкапсуляція, поліморфізм і успадкування, дає можливість користуватись всіма перевагами, які дає нам Java. Сама мова втілює багато кращих практик і шаблон дизайну в його бібліотеці. Java також сприяє використанню SOLID і Об’єктно-орієнтованих принципів проектування у вигляді проектів з відкритим вихідним кодом. Сама мова Java дуже проста. Проте Java поставляється з бібліотеки класів, яка надає найчастіше використовувані функції

утиліти, без яких більшості програм Java не обійтися. Ця бібліотека класів, яка називається Java API, є частиною Java як самої мови. Мова Java має лише 50 ключових слів, але Java API має кілька тисяч класів з десятками тисяч методів, які можна використовувати у ваших програмах. Вам не потрібно вивчати все API Java. Більшість програмістів знайомі з лише незначною його частиною. Якщо вам потрібно використовувати якийсь клас з API, з яким ви ще не знайомі, ви можете переглянути те, що робить клас у документації Java API. Розробка цією мовою дає нам можливість використовувати потужне середовище розробки таке як IntelliJ IDEA.

2.2.2 Середовище розробки IntelliJ IDEA

Середовище розробки IntelliJ IDEA – це інтегроване середовище розробки Java (IDE), яке користується великою популярністю серед розробників програмного забезпечення та компаній-виробників програмного забезпечення. Додаток завантажується з великим набором функцій та інструментів, які чудово підходять для розробки програмного забезпечення. IntelliJ IDEA розроблена з метою підвищення продуктивності шляхом надання найбільш інтуїтивної кодової допомоги для всіх підтримуваних мов. Середовище розробки IntelliJ IDEA надає функції налагодження для більшості додатків, які можуть підтримувати такі функції, як перегляд і маніпулювання об'єктами під час запуску коду. IDEA повністю підтримує розробку додатків Android. Необхідна мінімальна установка для того, щоб IDE було готове до використання. Великою перевагою є можливість зміни структури папок проекту на основі налаштувань, наприклад, ви можете змінити структуру з перегляду пакунками програми, а не файлами проекту. Характеристики середовища розробки IntelliJ IDEA:

- інструменти побудови;
- контроль версій;
- розумне завершення;
- аналіз потоку даних;

- ін'єкція мови;
- послідовне перетворення;
- виявлення дублікатів;
- швидкий пошук;
- евристика;
- декомпілятор;
- інструменти бази даних / SQL;
- відладчик;
- контроль версій тощо.

Середовище розробки IntelliJ IDEA дозволило нам використовувати Maven для створення нашого веб додатку.

2.2.3 Інструмент керування Maven

Інструмент керування Maven – це інструмент керування проектами, який надає розробникам повну структуру життєвого циклу. Команда розробників може автоматизувати побудову інфраструктури проекту практично в будь-який час, оскільки Maven використовує стандартну розкладку каталогів і життєвий цикл побудови за замовчуванням. У випадку декількох середовищ розробки команд, Maven може налаштувати спосіб роботи відповідно до стандартів за дуже короткий час. Оскільки більшість установок проекту є простими і багаторазовими, Maven робить життя розробника легким під час створення звітів, перевірок, побудови та тестування автоматизації.

Інструмент керування Maven надає розробникам можливості:

- просте налаштування проекту, що відповідає найкращим практикам;
- послідовне використання всіх проектів;
- управління залежностями, включаючи автоматичне оновлення.
- велике і зростаюче сховище бібліотек.

- легко писати плагіни на мовах Java або сценаріїв.
- миттєвий доступ до нових функцій з невеликою або без додаткової конфігурації.

Інструмент Maven спрощує і стандартизує процес створення проекту. Він обробляє компіляцію, розповсюдження, документацію, командну співпрацю та інші завдання. Maven підвищує можливості повторного використання і піклується про більшість завдань, пов'язаних з побудовою.

Основна мета Maven – надати розробнику наступні можливості:

- Комплексна модель для проектів, яка є багаторазовою, доступною для налагодження та легшою для розуміння.
- Додатки або інструменти, які взаємодіють з цією декларативною моделлю.

Структура і вміст проекту Maven оголошуються у файлі xml, який називається POM (Project Object Model), і який є основною одиницею всієї системи Maven. Це дозволило записати потрібні залежності до цього файлу, і не потрібно шукати бібліотеки з інтернету.

3 ПРОЕКТУВАННЯ СИСТЕМИ ОЦІНЮВАННЯ ІДЕНТИЧНОСТІ ТЕКСТІВ З УРАХУВАННЯМ СИНОНІМІЇ

Дослідження у сфері пошуку плагіату у текстових документах сфокусовані в трьох основних напрямках:

- класифікація програмних засобів пошуку плагіату [23; 24];
- аналіз існуючих алгоритмів пошуку ідентичних фрагментів тексту [12; 5] і розробка та опис нових методів пошуку плагіату [14; 22; 17; 27; 28; 21; 26];
- аналіз програмних засобів, що існують на ринку [13; 16; 19; 20; 3; 6; 10];
- розробка нових програмних засобів [1; 8; 13].

Було здійснено чимало спроб, щоб проаналізувати сучасні засоби пошуку плагіату у ряді робіт вітчизняних і зарубіжних вчених, а також на персональних блогах багатьох авторів та веб-сайтах. Обиралося до десяти варіантів ресурсів. Однак не було встановлено за якими критеріями був обраний той чи інший ресурс. На сьогоднішній день проведено досить мало повномасштабних досліджень, що порівнюють ефективність наявного інструментарію за єдиними критеріями та характеристиками, розділяючи їх на програмне забезпечення та онлайн сервіси.

В освіті та науці плагіат можливо зустріти трьох типів:

- текстовий плагіат – повне або часткове привласнення фрагментів тексту, повністю ідентичних або замінених синонімами, що присутній у наукових роботах, звітах, статях, тезах, кваліфікаційних робіт, тощо;
- плагіат програмних кодів – привласнення чужого програмного коду заради досягнення власних цілей, а також, що теж нерідко трапляється, заради видання чужого коду за власну розробку;
- плагіат в нетекстових джерелах – копіювання медіаресурсів (фото-, відеоматеріали, схеми чи діаграми, таблиці тощо).

Основною відмінністю між цими типами – це використання методів для пошуку плагіату у кожному з цих випадків.

Встановити наявність першого типу плагіату дещо важче, оскільки існує багато способів модифікувати початковий текст. Нині актуальними є дослідження в сфері внутрішнього пошуку плагіату в документі за умови відсутності списку джерел та на основі аналізу зміни авторського стилю в окремих частинах тексту (рис. 3.1).

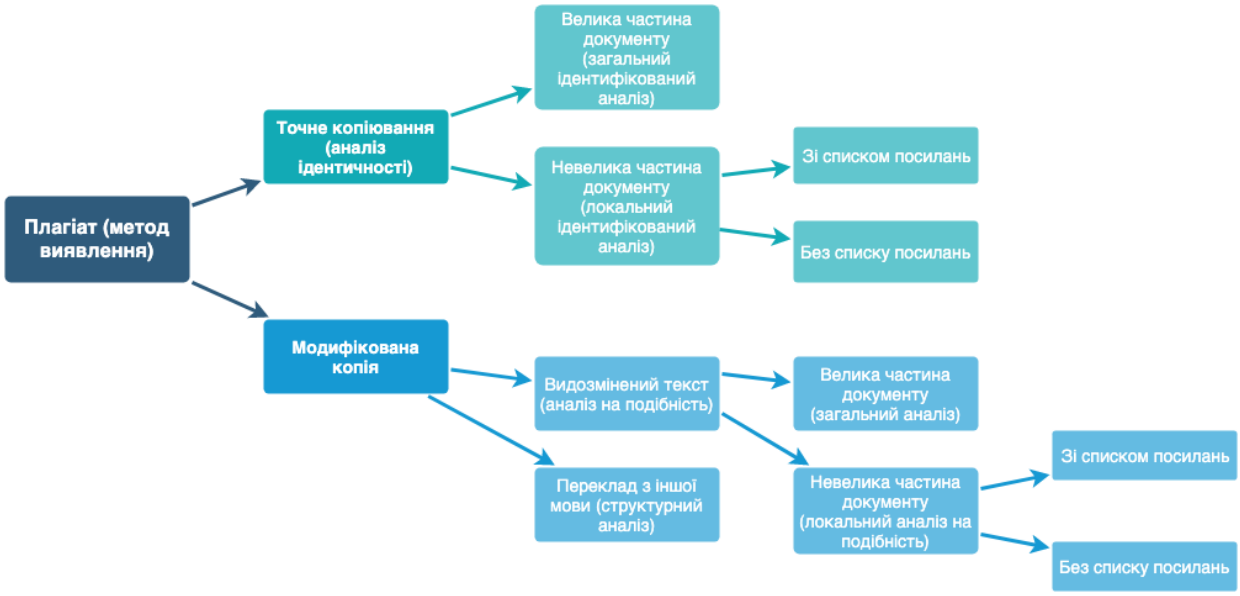

Рисунок 3.1 – Таксономія видів неправомірного запозичення текстів в поєднанні з відповідними їм методами пошуку плагіату

Існує два можливих шляхи щодо виявлення наявності плагіату:

- ручний пошук, що здійснюється безпосередньо викладачами, науковцями, редакторами, читачами журналів. Наявність в редакційній колегії наукових часописів добросовісних, чесних, свідомих рецензентів, що є експертами в своїй галузі та ґрунтовно аналізують рукописи, може значно мінімізувати проблему, однак не усунути її цілком;
- автоматичний пошук за допомогою комп’ютерної техніки та програмних засобів.

Під час дослідження було задіяно чимало сервісів для перевірки їх ефективності пошуку плагіату в науковії літературі, а також серед джерел в мережі Інтернет

(авторські блоги, статті тощо). Після проведення дослідження було проаналізовано та обрано низку кращих сервісів, які добре показали себе у ефективному пошуку плагіату у текстових документах. Список сервісів можна побачити у таблиці 3.1.

Таблиця 3.1

Сервіси для пошуку плагіату серед текстових документів		
Тип	Назва ресурсу	
Програмне забезпечення	Anti-Plagiarism	Praide Unique Content Analyser
	eTXT Антиплагиат	Viper
	Advego Plagiatus	Плагиата НЕТ
	Double Content Finder	PlagScan
	DupliChecker	Plagiarism Detector
	PaperRater	SeeSources
	Plagiarisma.net	FindCopy (Miratools)
Онлайн ресурси	PlagiarismChecker	Grammarly
	Plagium	Docoloc
	PlagTracker	Text.ru

4 РЕАЛІЗАЦІЯ СИСТЕМИ ОЦІНЮВАННЯ ІДЕНТИЧНОСТІ ТЕКСТІВ З УРАХУВАННЯМ СИНОНІМІЇ

4.1 Опис та тестування системи оцінювання ідентичності текстів з урахуванням синонімії

Основна ідея пошуку плагіату в роботі – це використання взаємодоповнюючих методів пошуку, що застосовуються до довільних текстів.

Процес виділення основних характеристик – це введення представлення, тобто з моделі, з великою кількістю надлишкової інформації, переходимо в більш компактну модель, де незначуща інформація видалена (нормалізація). Зазвичай в процесі нормалізації всі слова поділяються лише одним пропуском, і всі букви приводяться до єдиного регістру.

Вибираючи різні представлення, вибираються характеристики, які для цього випадку є основними і залишають їх. Після цього вводимо функцію близькості (метрику), щоб визначити, які характеристики з решти є більш, а які менш значимі. Те, які характеристики є основними – це питання підходу, питання розуміння плагіату. Метрики повинні представляти такі характеристики, які достатньо важко змінити, намагаючись замаскувати копію. Повинні бути стійкі до незначних з ним змін вихідного коду. Повинно бути можливо просто порівнювати, використовуючи ці метрики, і вони повинні бути досить загальними.

Процес роботи системи полягає у зчитуванні текстів з обраного файлу користувачем, а також файлів, які знаходяться у тій самій директорії. Наступним кроком тексти розбиваються на речення, а потім на слова. Отриманий список слів проходить відсіювання стоп-слів та службових символів й залишає лише головні слова. Далі для кожного головного слова формується список синонімів для подальшої перевірки текстів на плагіат та йде пошук послідовності слів та їх синонімів. Якщо

програма знайшла три та більше послідовних слів, цей фрагмент тексту помічається запозиченим.

Головне меню системи оцінювання ідентичності текстів з урахуванням синонімії (рис 4.1) складається с таких компонентів, як:

- поле для вводу шляху до файлу, який перевірятиметься на ідентичність;
- кнопка «Обрати», за допомогою якої користувач здатен обрати файл зручним для нього способом;
- поле для виводу інформації процесу оцінювання ідентичності текстів з урахуванням синонімії, на якому відображається список файлів, що будуть слугувати шаблонами для перевірки обраного файлу на ідентичність та відсоток запозиченості тексту обраного файлу по відношенню до інших файлів;
- прогрес, який вказує на відсоток виконаної роботи;
- кнопка «Перевірити», що запускає процес оцінювання ідентичності текстів з урахуванням синонімії;
- кнопка «Отримати звіт», яка відкриває текстовий файл з розгорнутою інформацією про перевірений на плагіат текст.

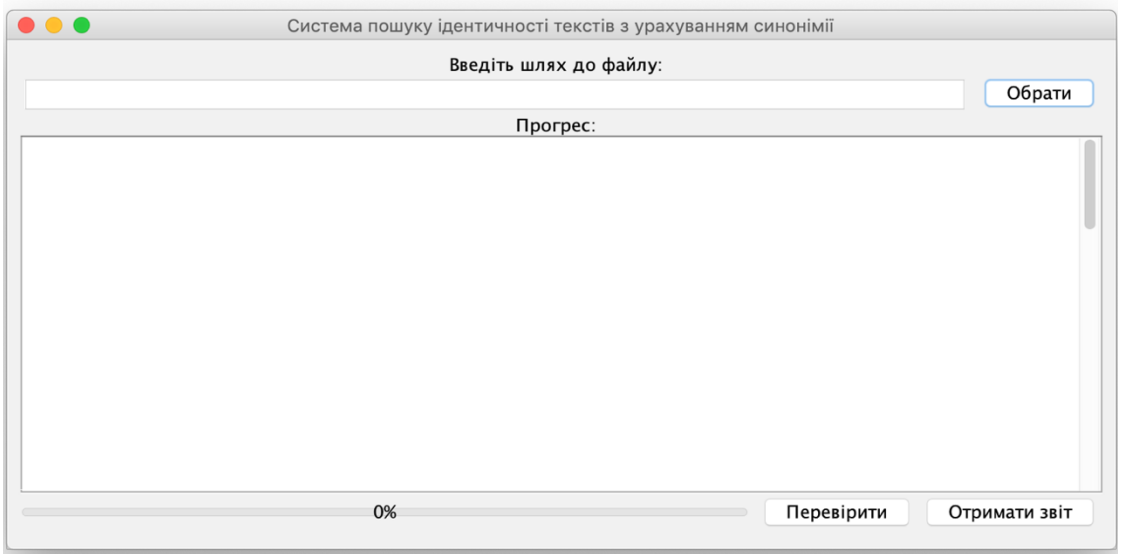

Рисунок 4.1 – Головне меню програми

Меню має мінімалістичний характер для того, щоб користувач концентрувався

лише на поставленій задачі, а саме – оцінюванні ідентичності текстів з урахуванням синонімії.

Після натискання на кнопку «Обрати» програма дає змогу обрати файл, який знаходиться локально на комп'ютері (рис. 4.2). Система має змогу читати текст з декількох форматів файлів, а саме:

- txt;
- doc;
- pdf.

Коли файл було обрано, користувач повертається до головного вікна із заповненим шляхом до файлу. Для того, щоб користувач запустив оцінку тексту на плагіат, йому потрібно натиснути на кнопку «Перевірити». Після цього він може слідкувати за процесом за допомогою інформаційного поля під назвою «Прогрес» та за повзунком з відсотком прогресу (рис. 4.3). Після закінчення оцінювання тексту на ідентичність з урахуванням синонімії користувача буде сповіщено за допомогою діалогового вікна з текстом "Завершено. Звіт сформовано у файлі під назвою Output.txt" (рис. 4.4).

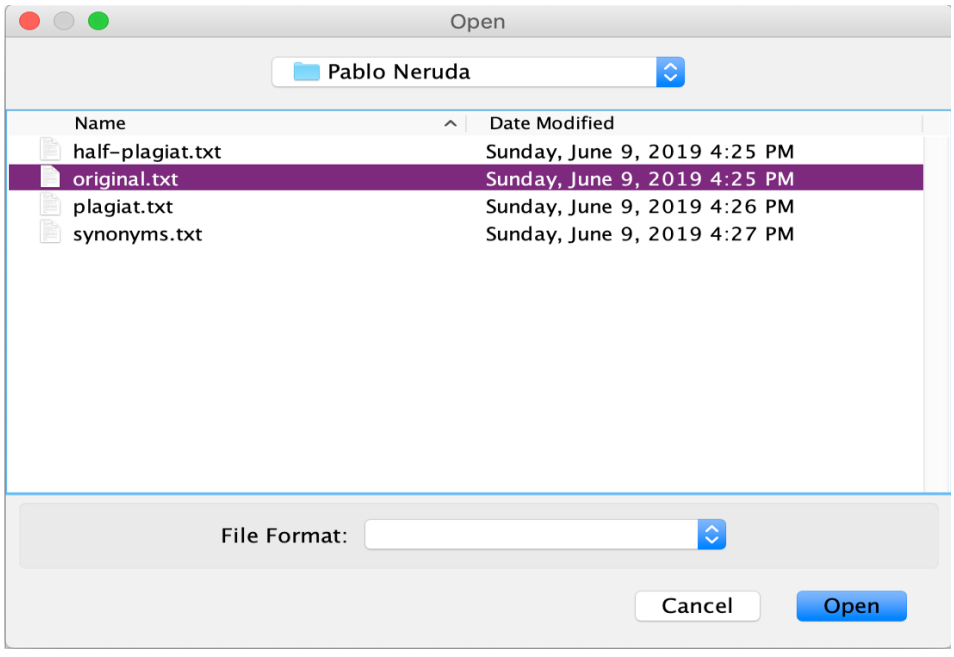

Рисунок 4.2 – Вікно вибору файлу

Алгоритм оцінювання ідентичності текстів з урахуванням синонімії:

- зчитування вхідних текстів з файлів, шлях до яких вказав користувач;
- розбиття зчитаних текстів на речення;
- виключення з речень службових символів, дублюючих пробілів та стоп-слів, залишаючи лише головні слова;
- розбиття кожного речення з головними словами та формування списку головних слів;
- формування списку синонімів для кожного слова з локального сховища та за їх відсутності отримання нових синонімів з інтернету;
- перевірка схожої послідовності головних слів та за наявності такої вважати цю послідовність слів запозиченою;
- отримання відсотку унікальності тексту (рис. 4.3).

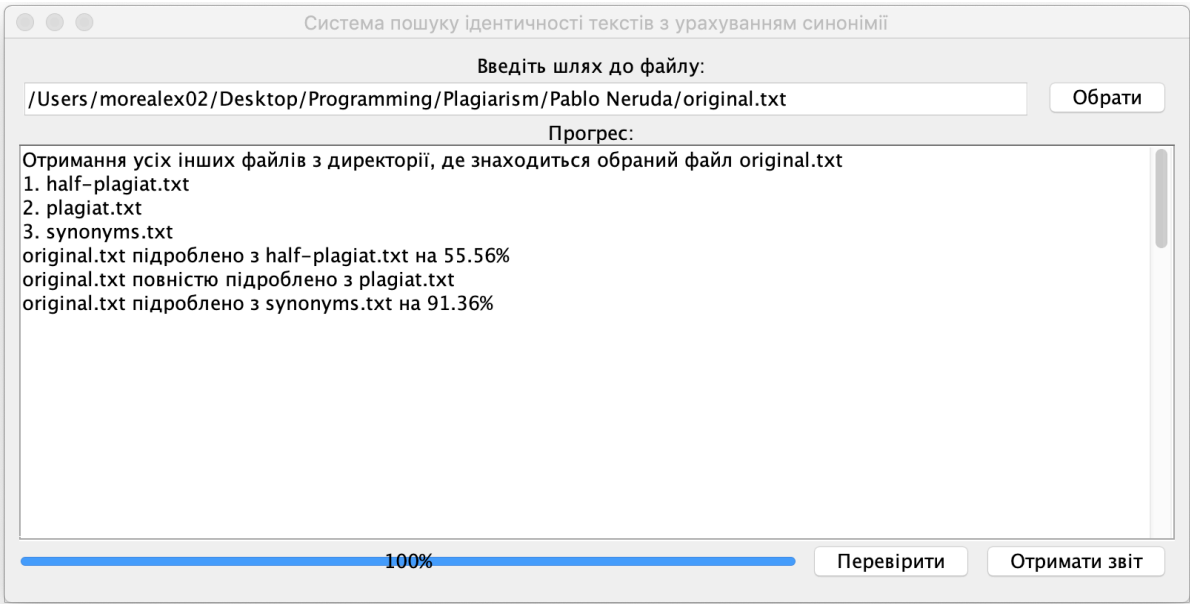

Рисунок 4.3 – Вікно головного меню із завершеною перевіркою на плагіат

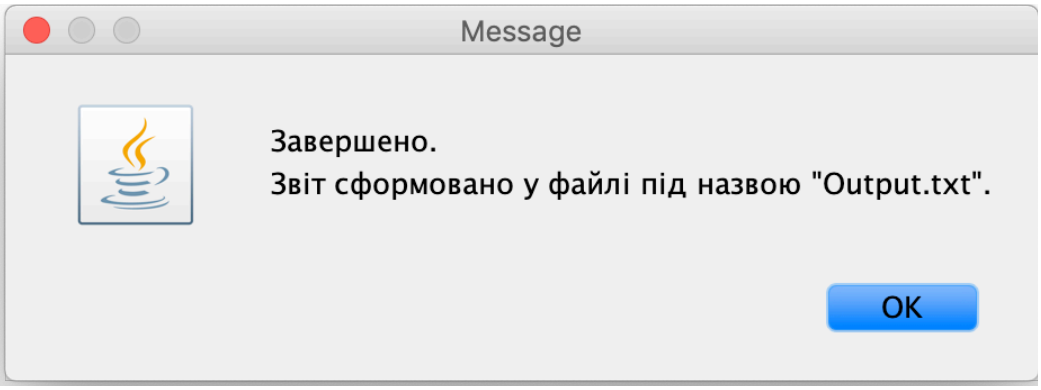

Рисунок 4.4 – Сповіщення про виконану роботу та створений звіт

Система оцінювання ідентичності текстів з урахуванням синонімії перед самим процесом видаляє стоп-слова (рис 4.5). Це потрібно для продуктивнішої перевірки текстів на плагіат, оскільки з текста вилучаються малозначущі слова та залишаються лише головні слова, які несуть у собі головну ідею. Для цього система має базовий список стоп-слів та перевіряє з цим списком вхідні тексти.

Рисунок 4.5 – Список збережених стоп-слів

4.2 Приклад роботи системи

Для тестування системи оцінювання ідентичності текстів з урахуванням синонімії за основу були взяті вірші:

- If You Forget Me [29];
- Phenomenal Woman [30].

Обраним текстовим файлом, який перевірявся на плагіат, був original.txt. Файл plagiat.txt був повністю скопійованим з original.txt. Файл half-plagiat.txt наполовину заміщений іншим віршем, а файл synonyms.txt має той самій текст, що і у original.txt, але замінений десятком синонімів.

Для початку користувачу потрібно було обрати файл для перевірки та натиснути на кнопку «Перевірити». Після цього користувач стає проінформованим про те, що система зчитує текст з файлу. Далі система починає зчитування з інших файлів, що знаходяться у тій самій директорії. Вони слугують ресурсом, з яким буде проходити оцінка запозиченості обраного файлу.

Після зчитування проходить процес нормалізації текстів з вилученням стоп-слів, формується список синонімів для кожного головного слова у тексті та починається процес оцінювання ідентичності текстів з урахуванням синонімії.

Коли процес оцінювання двох текстових файлів закінчився, користувач може побачити відсоток запозиченості. Файл original.txt, що оцінювався з іншими файлами, має наступні результати:

- вміст на 55.56% запозичено з файлу half-plagiat.txt;
- вміст повністю скопійовано з файлу plagiat.txt;
- вміст на 91.36% запозичено з файлу synonyms.txt;

4.3 Аналіз результатів роботи системи

Звіт оцінювання ідентичності текстів з урахуванням синонімії, що вказаний на рисунку 4.6, відображає:

- кількість використаних слів (це лише головні слова, оскільки стоп-слова було вилучено);
- використаних синонімів;
- відсоток запозиченості тексту;

Якщо текст був повністю скопійованим, звіт не стане виводити список синонімів та використаних слів, оскільки ця інформація є повністю вичерпною з вмісту запозиченого тексту.

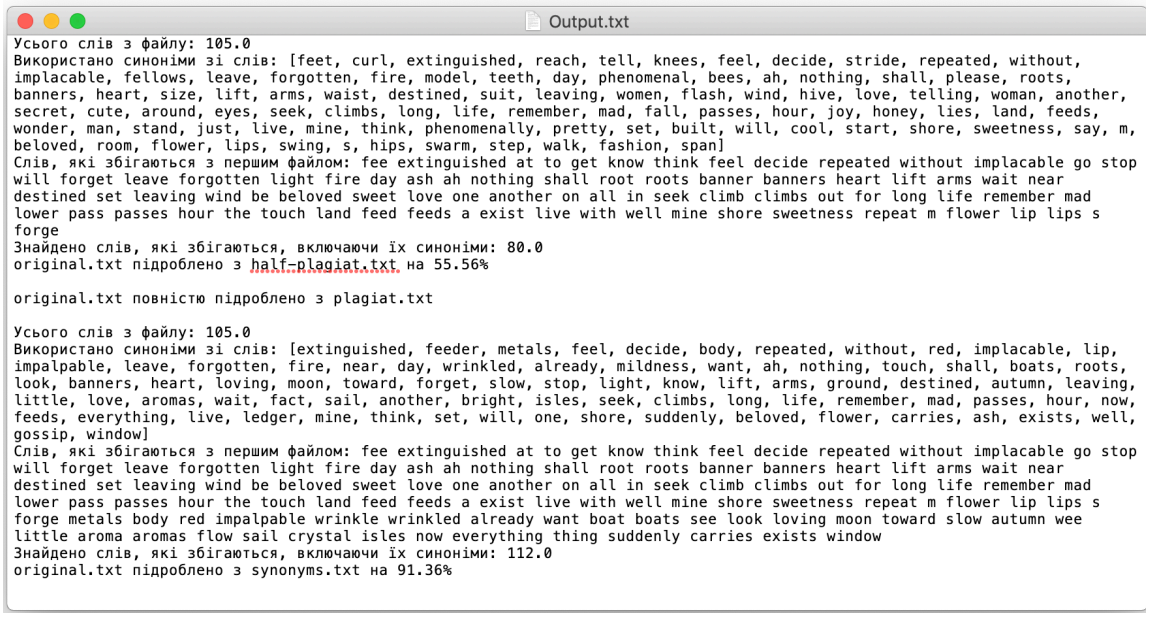

Рисунок 4.6 – Звіт оцінки текстів на ідентичність з урахуванням синонімії

Перевіривши результати оцінювання ідентичності текстів з урахуванням синонімії, система надала коректний відсоток, а саме:

- оригінальний текст на 55 відсотків запозичено з тексту у файлі half-plagiat.txt, оскільки цей файл мав лише половину вмісту, який був у текст, що оцінювався (рис 3.7);
- оригінальний текст повністю скопійовано з файлу plagiat.txt (рис 4.7 та 4.8)
- оригінальний текст на 91 відсоток запозичено з тексту у файлі synonyms.txt, що означає гарний результат роботи системи, оскільки задіяно більше десяти синонімів з вузькою спеціалізацією (рис 4.7).

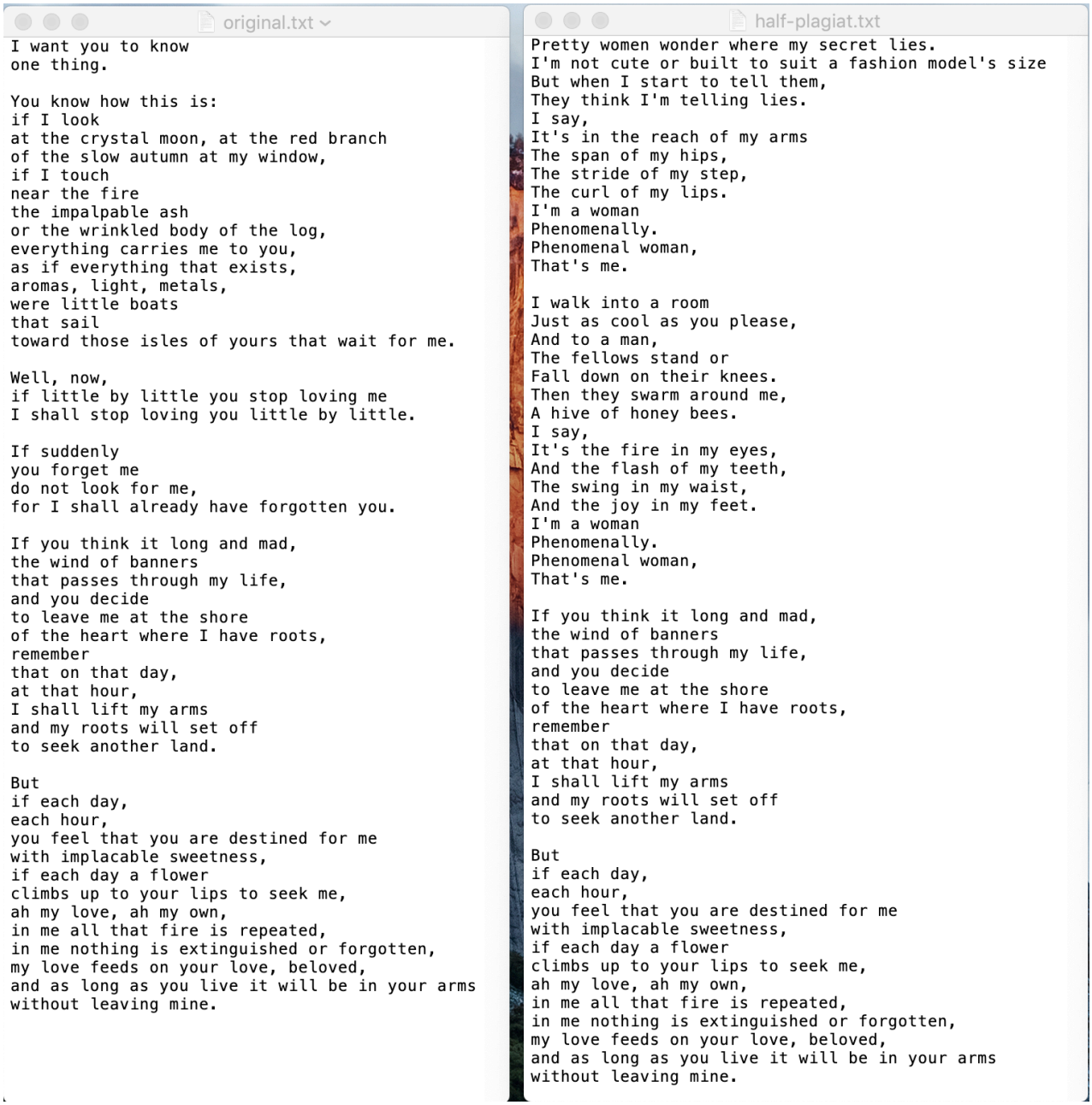

Рисунок 4.7 – Вміст текстових файлів original.txt та half-plagiat.txt

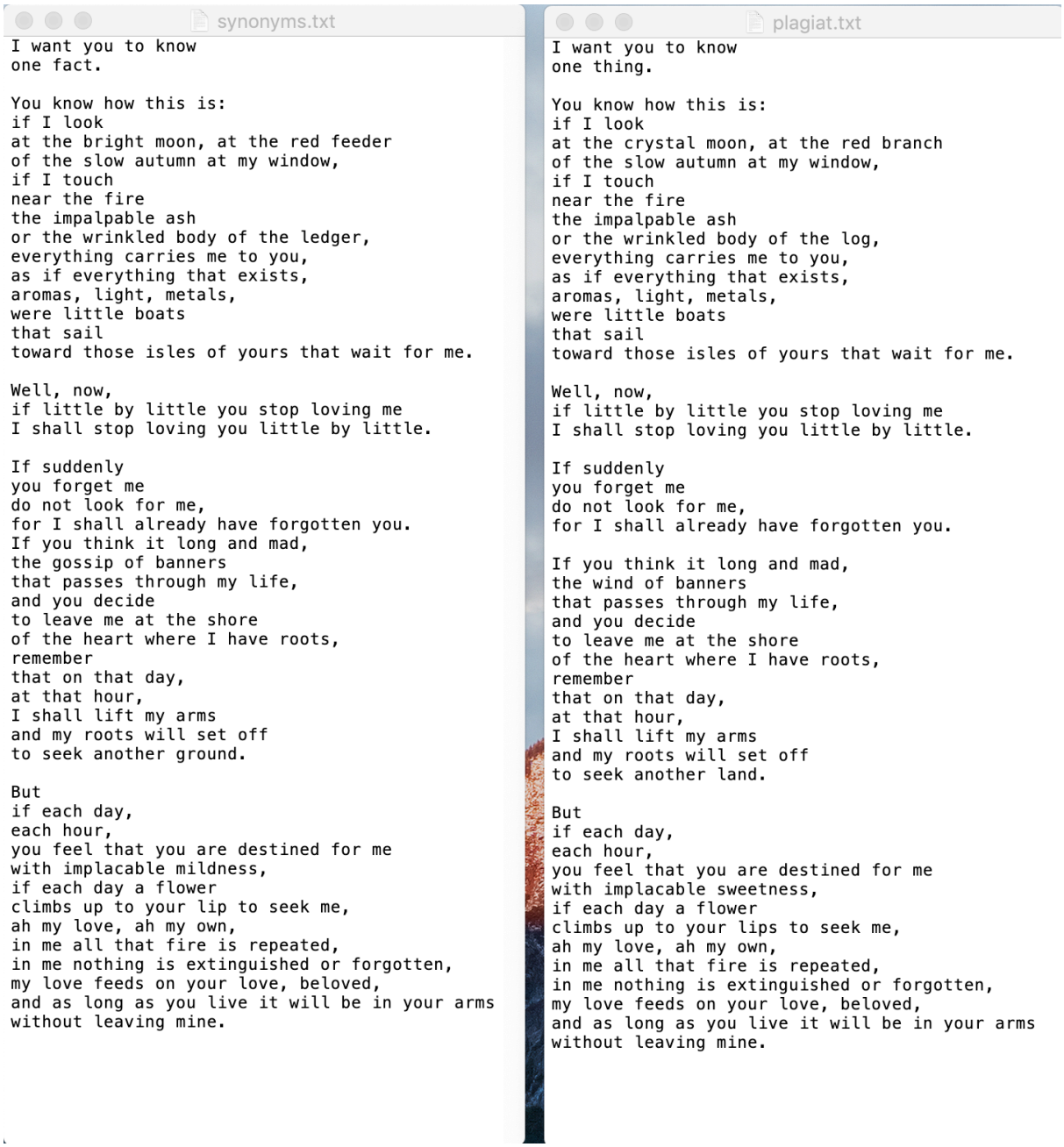

Рисунок 4.8 – Вміст текстових файлів synonyms.txt та plagiat.txt

ВИСНОВКИ

В результаті виконання дипломної роботи автором виконані наступні роботи і отримані наступні результати:

- удосконалено інструментальні засоби оцінки та аналізу ідентичності текстів з урахуванням синонімії;
- розробка набору інструментів для відсіювання стоп-слів та службових символів з текстів для їх коректного аналізу;
- реалізовано збереження системою списку синонімів заради зменшення витраченого часу на процес пошуку плагіату та можливістю доповнення існуючого списку за допомогою запитів до сервісу, який постачає синоніми;
- система може використовуватись як самостійно, так і модульно, тобто підключатися до інших систем, отримувати файли для перевірки їх на ідентичність та повертати звіт щодо їхнього аналізу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Болілий В.О. Перевірка унікальності тексту при оцінюванні студентських робіт творчого або дослідницького характеру/ В.О. Болілий, В. В. Копотій // Наукові записки НДУ ім. М. Гоголя. – 2011. – № 7 (34).— С. 134—145.
2. Квашина Ю. А. Методы поиска дубликатов скомпонованных текстов научной стилистики [Электронный ресурс] / Ю. А. Квашина // Технологический аудит и резервы производства. – 2013. – № 3 (3).— С. 16-20.
3. Котляров И. Д. Самоплагиат в научных публикациях. / И. Д. Котляров // Научная периодика: проблемы и решения. – 2007. – № 2.— С. 6-12.
4. Ліннік І. Програмне забезпечення для виявлення плагіату: практичний аспект [Електронний ресурс] / І. Ліннік // Науковий блоґ НаУ «Острозька Академія». – 2013.– Режим доступу: http://naub.oa.edu.ua/2013/prohramne-zabezpechennya-dlya-vyyavlennya-plahiatu-praktychnyj-aspekt/
5. Михайловський Ю.Б. Система Anti-Plagiarism як інструмент запобігання плагіату в навчальній та науковій діяльності / Ю.Б. Михайловський, Н.А. Длугунович // Вісник Хмельницького національного університету. Технічні науки. – 2013. – № 3.— С. 162—168.
6. Петренко В.С. Поняття та види плагіату. / В.С. Петренко // Часопис цивілістики. – 2013. – Вип. 14.— С. 128—131.
7. Поповський О. І. Огляд програм порівняльного аналізу на збіг / О. І. Поповський // Збірник тез доповідей 2-го Кіровоградського соціально-економічного форуму «Інформаційне суспільство і влада». – Кіровоград. – 2013. – С. 99—100.
8. Про авторське право і суміжні права: Закон України від 23.12.1993 № 3792-XII // Відомості Верховної Ради України від 29.03.1994 року. – № 13. – Ст. 64.
9. Чижова А. А. Алгоритми пошуку плагіату [Електронний ресурс] / А. А. Чижова // Східно-Європейський журнал передових технологій. – 2010. - 4, № 2

(46), С. 13-16.

13. Шарапова Е. В. “ Универсальная система проверки текстов на плагиат «Автор.net»” / Е. В. Шарапова, Р. В. Шарапов // Информатика и её применения – 2012. – № 3 (6).— С. 52–58.
14. Шинкаренко В. І. Система контролю плагіату в студентських роботах [Електронний ресурс] / В. І. Шинкаренко, О. С. Куроп’ ятник // Східно-Європейський журнал передових технологій. – 2012. – Том. 4. – № 2(58). – С. 32-36.
15. Энциклопедический словарь Ф.А. Брокгауза и И.А. Ефрона [Текст] / Ф.А. Блокгауз, И.А. Ефрон. - в 82 основ.и 4 допол. полутомах. - СПГ, 1898. - Т. XXIII, кн. 48. - 961 с.
16. Ali A. M. El Tahir Overview and Comparison of Plagiarism Detection Tools [online] / A. M. El T. Ali, H. M. D. Abdulla, V. Snasel// CEUR Workshop Proceedings. – 2011. – Vol. 706.— P. 161-172.– Режим доступу: http://ceur-ws.org/Vol-706/poster22.pdf
17. Anzelmi D. Plagiarism Detection Based on SCAM Algorithm [online] / D. Anzelmi, D. Carlone, F. Rizzello, R. Thomsen, D. M. Akbar Hussain// Proceedings of the International MultiConference on Engineers and Computer Scientists 2011. – 2011. – № 6 (1).— P. 272-277– Режим доступу: http://www.ijsce.org/attachments/File/v2i5/E0984092512.pdf
18. Berlinck R.G. S. The academic plagiarism and its punishments - a review / R.G. S. Berlinck // Brazilian Journal of Pharmacognosy. – 2011. - № 21(3) — P. 365-372.
19. Bull J. Technical Review of Plagiarism Detection Software Report [online] / J. Bull, E. Coughlin, C. Collins, D. Sharp – Luton. –2000. – P. 36-39.
20. Hunes C. Examining Anti-Plagiarism Software: Choosing the Right Tool [Електронний ресурс] / C. Hunes, J. Stiffler, M. Malsed. – Clermont. – 2003. – P. 60-63.
21. Hriharan Sh. Automatic Plagiarism Detection Using Similarity Analysis [online] / Sh. Hariharan // The International Arab Journal of Information Technology. – 2012. – №

4 (9).— P. 322-326 – Режим доступу: http://www.ccis2k.org/iajit/PDF/vol.9,no.4/2796-4.pdf

22. Kharat R. Semantically Detecting Plagiarism for Research Papers [online] / R. Kharat, P. M. Chavan, V. Jadhav, K. Rakibe // International Journal of Engineering Research and Applications (IJERA) – 2013. – № 3 (3).— P. 077-080 – Режим доступу: http://www.ije ra.com/papers/Vol3_issue3/P33077080.pdf
23. Lancaster, T. Effective and Efficient Plagiarism Detection : PhD thesis / Lancaster Thomas. – London, 2003. – P. 228-235.
24. Lancaster T. Classifications of Plagiarism Detection Engines [online] / T. Lancaster, F. Culwin // Innovation in Teaching and Learning in Information and Computer Sciences. 2005. – №2(4). – 16 p. – Режим доступу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.184.2406&rep=rep1&type=pdf
25. Marcial D. E. ICT Skills Enhancement Training in Teacher Education: The Case in Central Visayas, Philippines [online] / D. E. Marcial, M. S. Fortich, J. B. Rendal// Information Technologies and Learning Tools. – 2014. – № 1 (39).— P. 230-240 – Режим доступу: http://journal.iitta.gov.ua/index.php/itlt/article/view/964/749
26. Shenoy M. Automatic Plagiarism Detection Using SimilarityAnalysis [online] / M. Shenoy, K. C.Shet, U.

D. Acharya// Advanced Computing: An International Journal (ACIJ). – 2012. – № 3 (3).— P. 59-62 – Режим доступу: http://airccse.org/journal/acij/papers/0512acij06.pdf

27. Singh R. Duplicity Detection System for Digital Documents [online] / R. Singh, C. Dutta// International Journal of Soft Computing and Engineering (IJSCE). – 2012. – № 5 (2).— P. 24-28 – Режим доступу: http://www.iaeng.org/publication/IMECS2011/IMECS2011_pp272-277.pdf
28. Tschuggnall M. Detecting Plagiarism in Text Documents through Grammar-Analysis of Authors / M. Tschuggnall, G. Specht // 15th GI-Symposium Database Systems for

Business, Technology and Web, 11th March - 15th March, 2013. – 2013. — P. 241-259 – Режим доступу: http://www.btw-2013.de/proceedings/Detecting%20Plagiarism%20in%20Text%20Documents%20through%20Grammar Analysis%20of%20Authors.pdf

29. Pablo Neruda. If You Forget Me [Електронний ресурс] / Pablo Neruda – Режим доступу: https://www.poemhunter.com/poem/if-you-forget-me/
30. Maya Angelou. Phenomenal woman [Електронний ресурс] / Maya Angelou – Режим доступу: https://www.poetryfoundation.org/poems/48985/phenomenal-woman
31. Слесарева И. П. Проблемы описания и преподавания русской лексики / Слесарева И. П. // Системность языка – 2010. – № 3 (6).— С. 62–68.
32. Половникова В. И. Русский язык для иностранный студентов / Костомаров В. Г., Половникова В. И. // Лексические единицы – 2010. – № 2— С. 77–79.
33. Сорокин Ю. С. Развитие словарного состава русского литературного языка 30-90 годов XIX столетия / Сорокин Ю. С. // Семантические связи в словах – 1965. – № 2— С. 123–144.
34. Зиновьева Е. И. Лингвокультурология: теория и практика / Зиновьева Е. И., Юрков Е. Е. // Взаимосвязь языка и культуры – 2009. – № 5— С. 292-296.

Додаток А

Система оцінювання ідентичності текстів з урахуванням синонімії

Специфікація

УКР.НТУУ“КПІ”.ТР5165_19Б

Аркушів 2

2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ ім. Ігоря Сікорського».ТР5165_19Б 81-1	Записка	Пояснювальна записка
Компоненти		
УКР.НТУУ«КПІ».ТР5165_19Б 12-1	Текст програмного модуля	
УКР.НТУУ«КПІ».ТР5165_19Б 13-1	Опис програми	

Додаток Б

Система оцінювання ідентичності текстів з урахуванням синонімії

Текст програмного модуля

УКР.НТУУ“КПІ”.ТР5165_19Б 12-1

Аркушів 14

2019

```
package plagiarismDetecor;

import java.io.File;
import java.io.FileInputStream;
//import java.io.FilenameFilter;
import java.io.IOException;

import java.text.DecimalFormat;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.ListIterator;
import java.util.Set;
import java.util.logging.FileHandler;
import java.util.logging.Logger;

import javax.swing.JOptionPane;
import javax.swing.SwingWorker;

import org.apache.commons.io.FileUtils;
import org.apache.commons.lang.ArrayUtils;

import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.text.PDFTextStripper;

import org.apache.poi.hwpf.HWPFDocument;
import org.apache.poi.hwpf.extractor.WordExtractor;
import org.apache.poi.xwpf.usermodel.XWPFDocument;
import org.apache.poi.xwpf.usermodel.XWPFParagraph;
```

```
import services.DirectoryService;
import services.SynonymService;

public class PlagiarismDetector extends SwingWorker<Void, String> {

    private String randomNumber;
    private String fileName;
    private String directoryPath;
    private String[] common = null;

    private KMPMatcher matcher = new KMPMatcher();

    PlagiarismDetector(String fileName, String path) {
        this.fileName = fileName;
        this.directoryPath = path;
    }

    PlagiarismDetector(String path) {
        this.directoryPath = path;
    }

    private static void failIfInterrupted() throws InterruptedException {
        if (Thread.currentThread().isInterrupted()) {
            throw new InterruptedException("Interrupted while searching files");
        }
    }

    @Override
    protected Void doInBackground() throws Exception {
        failIfInterrupted();
        File dir = new File(directoryPath);

        if (dir.isDirectory()) {
```

```
            File[] txtFiles = dir.listFiles((dir1, filename) -> filename.endsWith(".txt"));
            File[] docFiles = dir.listFiles((dir12, filename) -> filename.endsWith(".doc"));
            File[] docxFiles = dir.listFiles((dir13, filename) -> filename.endsWith(".docx"));
            File[] pdfFiles = dir.listFiles((dir14, filename) -> filename.endsWith(".pdf"));
            File[] txtDocFiles = (File[]) ArrayUtils.addAll(txtFiles, docFiles);
            File[] txtDocDocxFiles = (File[]) ArrayUtils.addAll(txtDocFiles, docxFiles);
            File[] allFiles = (File[]) ArrayUtils.addAll(txtDocDocxFiles, pdfFiles);

            publish("Отримання усіх інших файлів з директорії, де знаходиться обраний файл " +
fileName);

            String[] listOfPaths = Arrays.stream(allFiles).map(File::getAbsolutePath)
                  .toArray(String[]::new);

            // Sorting list of paths
            List<String> tList = new ArrayList<>(Arrays.asList(listOfPaths));
            Collections.sort(tList);
            listOfPaths = tList.toArray(new String[tList.size()]);

            for (int i = 0; i < listOfPaths.length - 1; i++) {
               publish((i + 1) + ". " + DirectoryService.getFileName(listOfPaths[i]));
            }

      //          for (String names: listOfPaths) {
      //             publish(DirectoryService.getFileName(names));
      //          }

            if (listOfPaths.length < 2) {
               JOptionPane.showMessageDialog(null, "Будь ласка, оберіть файл, який знаходиться
серед хоча б одного іншого файлу для перевірки ідентичності");
            }

            for (int i = 0; i < listOfPaths.length; i++) {
```

```
                for (int j = 0; j < listOfPaths.length; j++) {
//                  if (i != j) {
                    if (i != j && DirectoryService.getFileName(listOfPaths[i]).equals(fileName)) {
                        checkPlagiat(listOfPaths[i], listOfPaths[j]);
                    }

                    int x = ((i * listOfPaths.length + j + 1)) * 100 / listOfPaths.length / listOfPaths.length;
                    String progress = Integer.toString(x);
                    publish(progress);
                }
            }
        }
        return null;
    }

    private void checkPlagiat(String filePath0, String filePath1) throws Exception {
        FileHandler handler = new FileHandler("plagiarism_logs.log", true);
        Logger logger = Logger.getLogger("plagiarismDetecor");
        logger.addHandler(handler);

        getStopWords();

        String f1 = readFile(filePath0);
        String f2 = readFile(filePath1);

        ArrayList<String> mainWordSentences1 = extractMainWords(splitLines(f1));
        ArrayList<String> mainWordSentences2 = extractMainWords(splitLines(f2));

        // Check if first text is equal to second one.
        if (mainWordSentences2.containsAll(mainWordSentences1)) {
            String print = DirectoryService.getFileName(filePath0) + " повністю підроблено з " +
DirectoryService.getFileName(filePath1);
            System.out.println(print);
```

```
            System.out.println();

            logger.info(print);
            publish(print);

            return;
        }

        String mainWordsWholeText1 = singleString(mainWordSentences1);
        String mainWordsWholeText2 = singleString(mainWordSentences2);

        ArrayList<String> uniqueMainWordsList1 = getUniqueWordsList(mainWordsWholeText1);
        ArrayList<String> uniqueMainWordsList2 = getUniqueWordsList(mainWordsWholeText2);

        HashMap<String, ArrayList<String>> synonymsFile2 =
getAllSynonyms(uniqueMainWordsList2); //synonymsFile2 has all the synonyms of the second file.

        /* ******************** Search ************************  */
        //now finalString contains all main words to be match.

        for (String wordFromList: uniqueMainWordsList2) {
            ArrayList<String> synonymList = synonymsFile2.get(wordFromList);

            for (String synonymWord: synonymList) {
                matcher.KMPSearch(synonymWord, mainWordsWholeText1);
            }
        }


        /* ************************ Calculate ***************************** */
        // counting total words from 2nd file.

        double totalNumberOfWords = countTotalWords(mainWordsWholeText1);
```

```
            System.out.println("Усього слів з файлу: " + (totalNumberOfWords));

            System.out.println("Використано синоніми зі слів: " + uniqueMainWordsList2);

            String result = singleString((ArrayList<String>) matcher.uniqueMatchedWords);
            System.out.println("Слів, які збігаються з першим файлом: " + result);

            double numberOfWordsMatched = countTotalWords(result);
            System.out.println("Знайдено слів, які збігаються, включаючи їх синоніми: " +
numberOfWordsMatched);


            /* ***************** Display Result ********************** */

            double plagiarismPercent = getPlagiarismPercent((ArrayList<String>)
matcher.uniqueMatchedWords, uniqueMainWordsList2, uniqueMainWordsList1);
            String print =  DirectoryService.getFileName(filePath0) + " підроблено з " +
                  DirectoryService.getFileName(filePath1) +
                        " на " + Double.parseDouble(new
DecimalFormat("##.##").format(plagiarismPercent)) + "%";
            System.out.println(print);
            System.out.println();
            logger.info(print);

            publish(print);

        }

        private void getStopWords() throws IOException {
            String fileWords = readFile("stopwords.txt");

            this.common = fileWords.split("\\r?\\n");
        }
```

```
    private String readFile(String path) throws IOException {
        String fileContent = null;

        if (path.endsWith(".txt")) {
            File file = new File(path);
            fileContent = FileUtils.readFileToString(file, "UTF-8");
        } else if (path.endsWith(".doc")) {
            fileContent = readDocFile(path);
        } else if (path.endsWith(".docx")) {
            fileContent = readDocxFile(path);
        } else if (path.endsWith(".pdf")) {
            fileContent = readPDFFile(path);
        } else {
            publish(path + " не підтримується. ");
        }

        return fileContent;
    }

    private ArrayList<String> splitLines(String fileContent) {
        fileContent = replaceRegex(fileContent);
        String[] lines = fileContent.split(randomNumber);

        return new ArrayList<>(Arrays.asList(lines));
    }

    private String replaceRegex(String contents) {
        String result;
        int rand = 3000 + (int) (Math.random() * 6000);
        this.randomNumber = " " + String.valueOf(rand) + " ";
        result = contents.replace(".", randomNumber).replace("?", randomNumber).replace("!",
randomNumber);
```

```
    return result;
}

private ArrayList<String> extractMainWords(ArrayList<String> file) {
    ListIterator<String> iterator = file.listIterator();

    while (iterator.hasNext()) {
        iterator.set(iterator.next()
                .replaceAll("[^0-9][.]|[.][^0-9]|(?![.,])\\p{Punct}|\\,", " ")
                .replace("\n", " ")
                .replace("\r", " ")
                .replaceAll("\\s+", " ")
                .toLowerCase());
    }

    for (int k = 0; k < file.size(); k++) {
        String[] words = file.get(k).split(" ");
        ArrayList<String> wordsList = new ArrayList<>(Arrays.asList(words));

        for (int i = 0; i < wordsList.size(); i++) {
            for (String aCommon: common) {
                if (wordsList.contains(aCommon)) {
                    wordsList.remove(aCommon);
                }
            }
        }

        StringBuilder sb = new StringBuilder();

        for (String s: wordsList) {
            sb.append(s);
            sb.append(" ");
```

```
            }

            file.set(k, sb.toString());
        }
        return file;
    }

    private String singleString(ArrayList<String> list) {
        StringBuilder sb = new StringBuilder();

        for (String s: list) {
            sb.append(s);
            sb.append(" ");
        }

        String result = sb.toString();
        result = result.replaceAll("\\s+", " ");

        return result;
    }

    private HashMap<String, ArrayList<String>> getAllSynonyms(List<String> words) throws
Exception {
        HashMap<String, ArrayList<String>> synonyms = null;

        SynonymService synonymService = new SynonymService(words);

        try {
            synonyms = synonymService.getSynonyms();
        } catch (IOException ignored) {}

        return synonyms;
    }
```

```
    private int countTotalWords(String str) {
        String trimmed = str.trim();

        return trimmed.isEmpty() ? 0 : trimmed.split("\\s+").length;
    }

    private double getPlagiarismPercent(ArrayList<String> matchedWords, ArrayList<String>
otherTextWords, ArrayList<String> allWords) {
        if (allWords.isEmpty()) {
            return 0.0;
        }

        int wordsCount = 0;

        for (String matchedWord: matchedWords) {
            if (otherTextWords.contains(matchedWord)) {
                wordsCount++;
            }
        }

        return 100.0 * wordsCount / allWords.size();
    }

    private String removeDuplicateWords(String file) {
        ArrayList<String> al = new ArrayList<>(Arrays.asList(file.split(" ")));
        Set<String> hs = new HashSet<>();
        hs.addAll(al);
        al.clear();
        al.addAll(hs);
        return singleString(al);
    }
```

```
private ArrayList<String> getUniqueWordsList(String file) {
    ArrayList<String> al = new ArrayList<>(Arrays.asList(file.split(" ")));
    Set<String> hs = new HashSet<>();
    hs.addAll(al);
    al.clear();
    al.addAll(hs);
    return al;
}

private String readDocFile(String path) {
    StringBuilder content = new StringBuilder();
    try {
        File file = new File(path);
        FileInputStream fis = new FileInputStream(file.getAbsolutePath());

        HWPFDocument doc = new HWPFDocument(fis);

        WordExtractor we = new WordExtractor(doc);
        String[] paragraphs = we.getParagraphText();
        for (String para: paragraphs) {
            content.append(para);
        }
        fis.close();
        return content.toString();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return content.toString();
}

private String readDocxFile(String path) {
    StringBuilder content = new StringBuilder();
    try {
```

```
        File file = new File(path);
        FileInputStream fis = new FileInputStream(file.getAbsolutePath());

        XWPFDocument document = new XWPFDocument(fis);

        List<XWPFParagraph> paragraphs = document.getParagraphs();

        for (XWPFParagraph para: paragraphs) {
            content.append(para.getText());
        }
        fis.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return content.toString();
}

private String readPDFFile(String path) throws IOException {
    String content = "";
    PDDocument pdDoc = null;

    try {
        pdDoc = PDDocument.load(new File(path));
        PDFTextStripper pdfStripper = new PDFTextStripper();
        pdfStripper.setStartPage(1);
        pdfStripper.setEndPage(pdDoc.getNumberOfPages());
        content = pdfStripper.getText(pdDoc);
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (pdDoc != null) {
            pdDoc.close();
        }
```

```
            }

            return content.replaceAll("\\r?\\n", " ");
        }

        @Override
        protected void process(List<String> chunks) {
            for (final String string: chunks) {
                if (string.matches("[0-9]+")) {
                    int progress = Integer.parseInt(string);
                    GUI.pbar.setValue(progress);
                } else {
                    GUI.processArea.append(string);
                    GUI.processArea.append("\n");
                }
            }
        }

        @Override
        protected void done() {
            JOptionPane.showMessageDialog(null, "Завершено.\nЗвіт сформовано у файлі під назвою
\"Output.txt\".");
        }
    }

    class KMPMatcher {

        List<String> matchedWords = new ArrayList<>();
        List<String> uniqueMatchedWords = new ArrayList<>();

        void KMPSearch(String word, String text) {
            int wordLength = word.length();
            int textLength = text.length();
```

```
int lps[] = new int[wordLength];
int i = 0;
int j = 0;

computeLPSArray(word, wordLength, lps);

List<Integer> position = new ArrayList<>();

while (i < textLength) {
    if (word.charAt(j) == text.charAt(i)) {
        j++;
        i++;
    }

    if (j == wordLength) {
        int pos = i - j;

        if (!position.contains(pos)) {
            position.add(pos);
            matchedWords.add(word);

            if (!uniqueMatchedWords.contains(word)) {
                uniqueMatchedWords.add(word);
            }
        }

        j = lps[j - 1];
    } else if (word.charAt(j) != text.charAt(i)) {
        if (j != 0) {
            j = lps[j - 1];
        } else {
            i = i + 1;
```

```
            }
        }
    }
}

private void computeLPSArray(String word, int wordLength, int lps[]) {

    int len = 0;
    int i = 1;
    lps[0] = 0;

    while (i < wordLength) {
        if (word.charAt(i) == word.charAt(len)) {
            len++;
            lps[i] = len;
            i++;
        } else {
            if (len != 0) {
                len = lps[len - 1];
            } else {
                lps[i] = len;
                i++;
            }
        }
    }
}
}
```

Додаток В

Система оцінювання ідентичності текстів з урахуванням синонімії

Опис програмного модуля

УКР.НТУУ“КПІ”.ТР5165_19Б 13-1

Аркушів 5

2019

АНОТАЦІЯ

Метою роботи було створення системи оцінювання ідентичності текстів з урахуванням синонімії. Програма забезпечує швидкий доступ до файлів з текстами для перевірки ідентичності, формування та ведення списку стоп-слів, пошук списку синонімів для кожного окремого слова, пошук ідентичності текстів з урахуванням синонімії. Розроблений програмний продукт може бути використаний, наприклад, в організаціях та установах, пов’язаних з науковою діяльністю.

ЗМІСТ

1 ВІДОМОСТІ ПРО ПРОГРАМНИЙ МОДУЛЬ

Даний програмний модуль розроблено у середовищі IntelliJ IDEA, використовуючи типізовану мову програмування Java та деякі додаткові бібліотеки як, наприклад. Maven.

Програма призначена для оцінювання ідентичності текстів з урахуванням синонімії.

1.1. Опис логічної структури

Було розроблено багатоплатформний програмний продукт, основною задачею якого є оцінювання ідентичності текстів. з урахуванням синонімії. При пошуку запозичених фрагментів текстів системи перевіряють на наявність послідовності ідентичних слів та враховують це за плагіат. Але сучасні системи не передбачають того, що слова можуть бути замінені їх синонімами. В такому випадку система не буде здатна відшукати запозиченість цього фрагменту тексту, оскільки не знайде послідовності ідентичних слів. Цей додаток має значну перевагу перед сучасними та наявними системами, оскільки має змогу знаходити послідовність фрагментів текстів за допомогою синонімії. Функціональні можливості додатку також включать в себе:

- збереження списку слів-синонімів локально та завантаження нових слів за допомогою доступу до інтернет-сервісів, які надають актуальний список синонімів;
- вилучення стоп-слів та службових символів з усіх зчитаних текстів для ефективної їхньої оцінки;
- відображення звіту щодо оцінювання тексту на ідентичність з іншими ресурсами з урахуванням синонімії.

Розроблена програма має коректний механізм оцінки ідентичних текстів з урахуванням синонімії. Завдяки цьому така система може бути використана в сфері, де потрібна перевірка вхідних текстів на плагіат наукових робіт для визначення унікальності інформації, яку хоче запропонувати автор.

1.2. Вхідні та вихідні дані

Вхідними даними для системи є текстові файли, один з яких оцінюватиметься на відсоток запозиченості, та решта інших, які виступають в ролі ресурсу, з яким буде перевірятися вхідний текст.

Вихідними даними є звіт про оцінювання ідентичності текстів з урахуванням синонімії.

2 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Програмний модуль було протестовано на лептопі з операційною системою macOS Mojave 10.14.4, який працює на базі процесору 2,6 GHz Intel Core i7 та має 16 Гб оперативної пам’яті. Розроблене програмне забезпечення є кросплатформенним, що дозволяє запускати його на комп’ютерах будь-якої потужності.